Dynamic memory access monitoring based on tagged memory

Michail Gorelov, Lev Mukhanov, Moscow Institute of Physics and Technology/MCST

Abstract

Software vulnerabilities become one of the top threats to world security in the coming decade. The most of such vulnerabilities are based on memory leaks and memory corruption. Many memory access monitoring tools exist, but most of them suffer from high overhead what makes it impossible to use such tools in the "real world" software projects. The goal of this research is to develop and investigate a memory access monitoring tool that detects memory leaks and corruption using tagged memory without reduce of an original application performance.

1 Introduction

Memory corruption bugs remain a serious problem for type-unsafe languages such as C or C++. For example, such vulnerabilities could be used for launching denial-of-service or cyber attacks [3].

There are several ways to prevent security breaches from such vulnerabilities. One way is using type-safe languages as Java. But usually such languages are not used for developing time-critical software because of poor performance. Another way is using static tools to detect memory access bugs. But unfortunately these tools generate many false positive and false negative errors [2]. Dynamic memory monitoring tools are much more effective than the static tools [1]. Such tools succeed in detecting a wide range of memory access errors but incur high overhead. For example, "Address Sanitizer" (the most advanced dynamic tool used by Google) detects memory corruption bugs at cost of 73% slowdown and 3.4x increased memory usage [4].

In this research we present a novel approach to dynamic memory access monitoring based on tagged memory. This approach makes it possible to avoid using of shadow metadata (shadow memory) and therefore to reduce overhead significantly. Instead of shadow memory we propose to use tags that contain information about correctness of accessed data. In other words we suggest that data from redzones would contain a specific(an incorrect) tag and data from "good" regions would contain a correct tag. Each attempt to access to data with an incorrect tag will cause a program to crash. Therefore there is no need to build for each memory access of a program an additional shadow memory access. An algorithm of redzones marking in stack, heap, global objects or dynamic memory is similar to "Address Sanitizer" algorithm. It should be mentioned that our approach can be applied to tagged architectures only.

For implementation and experiments we are going to use Elbrus architecture(a VLIW architecture with tagged memory). In Elbrus architecture tags are used to detect incorrect data during speculative execution of instructions. This is why it would be relatively easy to fit tags in the dynamic memory access monitoring tool for this architecture. Now we are implementing the code instrumentation phase in Elbrus optimizing compiler (stack protection has been supported already). Moreover we hope to apply this approach to our X86 binary compiler what will allow us to detect memory access bugs in native X86 applications without special recompilation.

The key contribution of this work is a new approach that can eliminate dynamic memory access overhead in real-world applications. We propose the use of tagged memory instead of shadow memory to control correctness of accessed data. Such approach will allow users to run any application(even real time applications or OS) with dynamic memory access monitoring without slowdown. Moreover we intend to implement this monitoring tool in Elbrus X86 binary compiler on the same principle.

References

[1] D. Bruening and Q. Zhao. Practical memory checking with dr. memory. In *Proceedings of the 9th Annual IEEE/ACM International Symposium on Code Generation and Optimization*, CGO '11, pages 213–223, Washington, DC, USA, 2011. IEEE Computer Society.

[2] D. L. Heine and M. S. Lam. A practical flow-sensitive and context-sensitive c and c++ memory leak detector. In *Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation*, PLDI '03, pages 168–181, New York, NY, USA, 2003. ACM.

[3] F. Qin, S. Lu, and Y. Zhou. Safemem: Exploiting ecc-memory for detecting memory leaks and memory corruption during production runs. In *In Proceedings of the 11th International Symposium on High-Performance Computer Architecture*, 2005.

[4] K. Serebryany, Bruening, A. Potapenko, and D. Vyukov. Addresssanitizer: A fast address sanity checker. In *USENIX ATC 2012*, 2012.